Sequencing, Alignment and Assembly

Shaun Jackman
Genome Sciences Centre
of the BC Cancer Agency
Vancouver, Canada
2011-July-14

Outline

- DNA sequencing
- Sequence alignment
- Sequence assembly
- Running ABySS
- Assembly visualization
- Transcriptome assembly and visualization

DNA sequencing technologies

- Sanger
- 454 Life Sciences
- Illumina
- SOLiD
- Ion Torrent
- Pacific Bio
- Helicos

Colour space

- Each colour call interrogates two nucleotides
  CCGATCATCACTGTATCTT
  0323213211211332 20
- A single-nucleotide variant affects two consecutive colour-space calls
  CCGATCATCAATGTATCTT
  0323213210311332 20
- A read that differs from the reference by a single colour-space call is a read error

Sequence alignment

Sequence alignment

- Global sequence alignment
  Needleman–Wunsch
- Local sequence alignment
  Smith–Waterman
- Glocal sequence alignment

The term glocal is a portmanteau of global and local.

Global alignment

- Full-length alignment of two sequences allowing for both mismatches and gaps
- Example:
  AGAGTGCTGCCGCC
  AGATGTACTGCGCC
- Alignment:

```
AGA-GTGCTGCCGCC
|||  ||  ||||  |||
AGATGTACTGC-GCC
```

- 12 matches of 15 bp alignment = 80% identity

Local alignment

- Find a matching substring of two sequences
- Unmatched bases are not penalized
- Example:
  ACGTAGATGTGCTGCCGCCACGT
  TTTGTACTGAAA
- ACGTAGATGTGCTGCCGCCACGT
  ```
          |||  |||
     TTTGTACTGAAA
  ```
- 6 matches of 7 bp = 86% identity

Glocal alignment

- Find a substring of the reference sequence that matches the entirety of the query sequence
- Example:
  Reference: ACGTAGATGTGCTGCCGCCACGT
  Query: TTTGTACTGAAA
- Alignment:

```
ACGTAGATGTGCTGCCGCCACGT
     |||  |||
   TTTGTACTGAAA
```

- 6 matches of 12 bp = 50% identity

Criteria for choosing an aligner

- Global, local or glocal alignment
- Aligning short sequences to long sequences such as short reads to a reference
- Aligning long sequences to long sequences such as long reads or contigs to a reference
- Handles small gaps (insertions and deletions)
- Handles large gaps (introns)
- Handles split alignments (chimera)
- Speed and ease of use

Short read aligners

- Bowtie
- BWA
- GSNAP

Short read aligners

Aligner	Purpose
Bowtie	Fast
BWA	Small gaps (indels)
GSNAP	Large gaps (introns)

Long sequence aligners

- BLAST
- BLAT
- BWA-SW
- Exonerate
- GMAP
- MUMmer

Long sequence aligners

Aligner	Purpose
BLAST	Many reference genomes
BLAT	Large gaps (introns)
BWA-SW	Small gaps (indels)
Exonerate	Ease of use
GMAP	Large gaps (introns)
MUMmer	Align two genomes

Heuristic Alignment
Seed and Extend

- For large sequences, an exhaustive alignment is very slow
- Many aligners start by finding perfect or near perfect matches to seeds
- The seeding strategy has a large effect on the sensitivity of the aligner
- BLAT for example requires two perfect nearby 11-mer matches

Sequence assembly

Source of DNA

- Genome
- Exome
- Clones and amplicons
- Transcriptome

Assembly

- Reference-based assembly
- *de novo* assembly

Assembly Algorithms

- Greedy
- Overlap, layout, consensus
- De Bruijn Graph or k-mer
- Burrows Wheeler transform and FM-Index

Greedy

- Find two sequences with the largest overlap and merge them; repeat
- Flaw: prone to misassembly

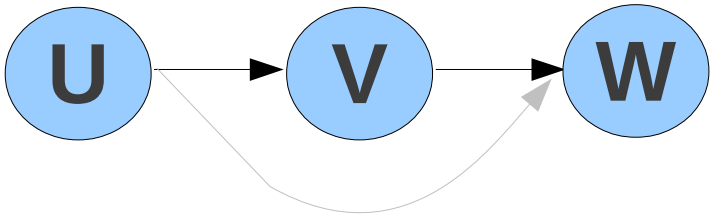
Overlap, Layout, Consensus

- **Overlap**
  Find all pairs of sequences that overlap
- **Layout**
  Remove redundant and weak overlaps.
  Merge pairs of sequences that overlap unambiguously; that is, pairs of sequences that overlap only with each other and no other sequence
- **Consensus**
  Call the consensus base at positions where reads overlap

Overlap graph

- A vertex is a string
- An edge represents an overlap between two strings
- Used by Overlap-Layout-Consensus assemblers

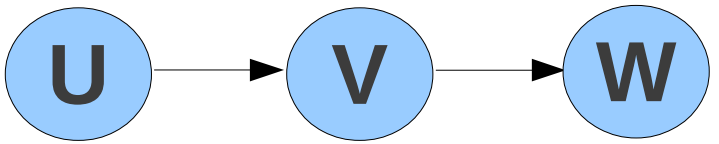
```
U AGATGTGCTGCCGCC
V      TGCTGCCGCCTTGGA
W           GCCGCCTTGGAAGAT
```



De Bruijn Graph

- A De Bruijn Graph is a particular kind of overlap graph
- A vertex is a string of length *k*
- An edge is an overlap of exactly length *k-1*
- Used by De Bruijn Graph assemblers

```
U AGATGTGCTGCCGCCTTGGAAG
V  GATGTGCTGCCGCCTTGGAAGA
W   ATGTGCTGCCGCCTTGGAAGAT
```



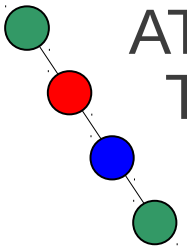
De Bruijn Graph

- For each read of length l, (l - k + 1) k-mers are generated by sliding a window of length k over the read

Read (l = 12):

ATCATACATGAT

k-mers (k = 9):

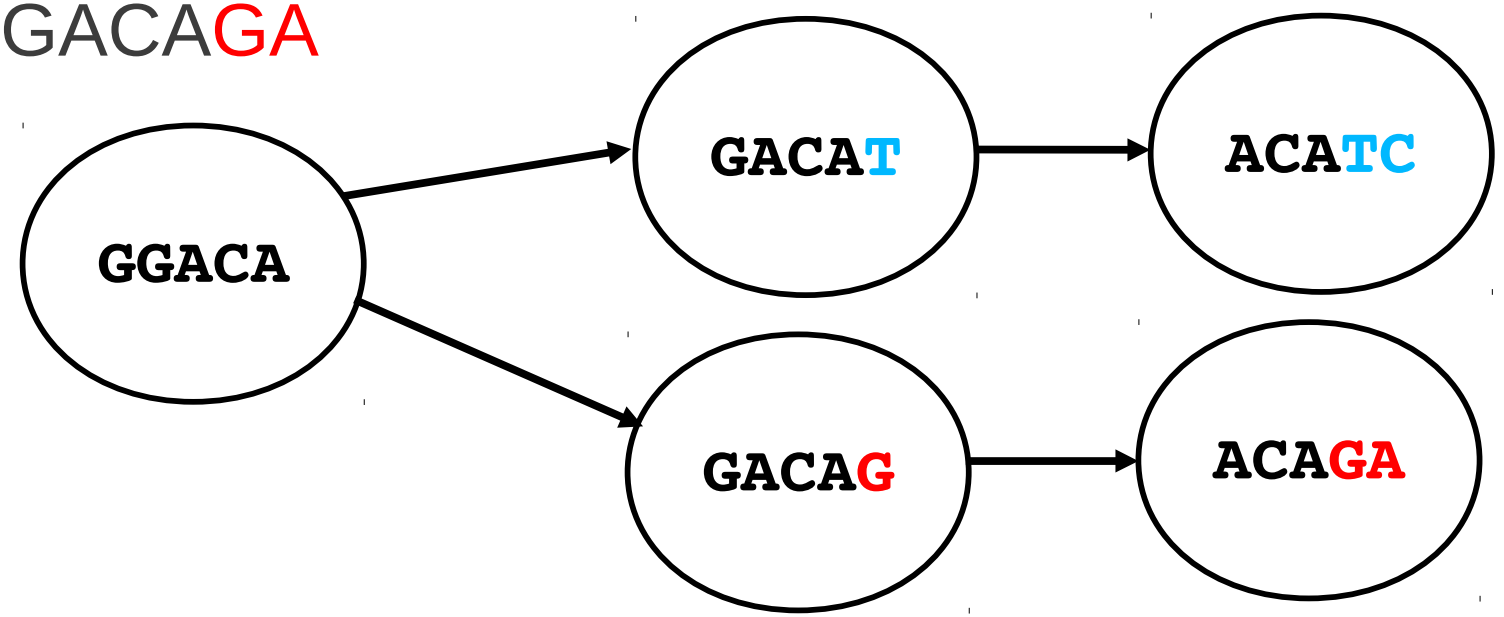


ATCATACAT
TCATACATG
CATACATGA
ATACATGAT

- Each k-mer is a vertex of the de Bruijn graph
- Two adjacent k-mers are an edge of the de Bruijn graph

De Bruijn Graph

- A simple graph for $k = 5$
- Two reads
  - GGACATC
  - GGACAGA



Burrows-Wheeler transform and the Ferragina-Manzini index

- A return to Overlap, Layout, Consensus
- Uses the BWT and FM-index to find all the pairs of overlapping sequences efficiently
- Typically requires less memory than a de Bruijn graph assembler

Assembly Algorithms

Algorithm	Purpose
OLC	Small genome Long reads Handles indels
De Bruijn graph	Large genome Short high-quality reads No indels
BWT	Large genome Short or long reads No indels (currently)

Overlap, Layout, Consensus

- ARACHNE
- CAP3
- Celera assembler
- MIRA
- Newbler
- Phrap

De Bruijn Graph

- ABySS
- ALLPATHS
- SOAP de novo
- Velvet

Burrows Wheeler Transform

- String Graph Assembler (SGA)

ABySS

- de Bruijn graph assembler
- Strengths
  - small memory foot print
  - distributed processing using MPI
  - can handle very large genomes

Velvet

- de Bruijn graph assembler
- Strengths
  - can use paired-end and mate-pair libraries
  - can mix short and long reads
  - can use a reference genome

SGA

- Overlap-graph assembler using the BWT
- Strengths
  - small memory foot print
  - can mix short and long reads
  - can handle large genomes

Sequencing strategy for assembly

- Reads are not sampled uniformly from the genome
- With short reads, 30x coverage is a minimum, 50-100x coverage is better, and even more is better still
- Quality filtering and trimming is treacherous, but can reduce assembly time and memory use
- Construct multiple libraries
- Use multiple sequencing technologies
- Hierarchical sequencing

Assembling to find variants

Small deletion in a tandem repeat

- The reference has 5 repetitions of a short 7-base sequence: GGCTGGA
- The sample has only 4 repetitions, one fewer

```
        Sample
0006813 TCCAAAT.......ggctggaggctggaggctggaggctggaggcATGTGTTAGTG 0006861
>>>>>>> |||||||       ||||||||||||||||||||||||||||||||||||||||| >>>>>>>
2356747 TCCAAATggctggaggctggaggctggaggctggaggctggaggcATGTGTTAGTG 2356802
        Reference
```

Alignment of short reads may not show the deletion

- Aligning reads to the reference perfectly covers the reference with no more than 2 errors per read
- Alignment will not find the small 7-base deletion

Reference:

```
TCCAAATggctggaggctggaggctggaggctggaggctggaggcATGTGTTAGTG
```

Alignment:

```
TCCAAATGGCTGGAGGCTGGAGGCTGGAGGCTGGAGG
 CCAAATGGCTGGAGGCTGGAGGCTGGAGGCTGGAGGC
  CAAATGGCTGGAGGCTGGAGGCTGGAGGCTGGAGGCA
   AAATGGCTGGAGGCTGGAGGCTGGAGGCTGGAGGCAT
    AATGGCTGGAGGCTGGAGGCTGGAGGCTGGAGGCATG
            ATGGCTGGAGGCTGGAGGCTGGAGGCTGGAGGCATGT
             TGGCTGGAGGCTGGAGGCTGGAGGCTGGAGGCATGTG
              GGCTGGAGGCTGGAGGCTGGAGGCTGGAGGCATGTGT
               GCTGGAGGCTGGAGGCTGGAGGCTGGAGGCATGTGTT
                CTGGAGGCTGGAGGCTGGAGGCTGGAGGCATGTGTTA
                 TGGAGGCTGGAGGCTGGAGGCTGGAGGCATGTGTTAG
                  GGAGGCTGGAGGCTGGAGGCTGGAGGCATGTGTTAGT
                   GAGGCTGGAGGCTGGAGGCTGGAGGCATGTGTTAGTG
```

Assembly clearly shows the deletion

- Assembling the reads and aligning the resulting contig to the reference clearly shows the small 7-base deletion.

```
Reads:  TCCAAATGGCTGGAGGCTGGAGGCTGGAGGCTGGAGG
         CCAAATGGCTGGAGGCTGGAGGCTGGAGGCTGGAGGC
          CAAATGGCTGGAGGCTGGAGGCTGGAGGCTGGAGGCA
           AAATGGCTGGAGGCTGGAGGCTGGAGGCTGGAGGCAT
            AATGGCTGGAGGCTGGAGGCTGGAGGCTGGAGGCATG
             ATGGCTGGAGGCTGGAGGCTGGAGGCTGGAGGCATGT
              TGGCTGGAGGCTGGAGGCTGGAGGCTGGAGGCATGTG
               GGCTGGAGGCTGGAGGCTGGAGGCTGGAGGCATGTGT
                GCTGGAGGCTGGAGGCTGGAGGCTGGAGGCATGTGTT
                 CTGGAGGCTGGAGGCTGGAGGCTGGAGGCATGTGTTA
                  TGGAGGCTGGAGGCTGGAGGCTGGAGGCATGTGTTAG
                   GGAGGCTGGAGGCTGGAGGCTGGAGGCATGTGTTAGT
                    GAGGCTGGAGGCTGGAGGCTGGAGGCATGTGTTAGTG
Contig: TCCAAATGGCTGGAGGCTGGAGGCTGGAGGCTGGAGGCATGTGTTAGTG

Alignment:
0006813 TCCAAAT.......ggctggaggctggaggctggaggctggaggcATGTGTTAGTG 0006861
>>>>>>> |||||||       |||||||||||||||||||||||||||||||||||||||| >>>>>>>
2356747 TCCAAATggctggaggctggaggctggaggctggaggctggaggcATGTGTTAGTG 2356802
```

Running ABySS

Stages of ABySS

- Assemble read sequences without paired-end information and output an initial assembly
- Map the reads back to the initial assembly
- Use the paired-end information to merge contigs from the first stage into larger sequences and output the final assembly

Memory requirement of ABySS

- Depends on the size of the genome, depth of coverage, quality of the reads, read length and the parameter *k*
- Difficult to predict in advance, but consistent between replicates
- ABySS can distribute the assembly over multiple machines and use the memory (RAM) of multiple machines
- Quality trimming can reduce memory usage

Example memory usage of ABySS

Genome size	RAM
200 kbp	¼ GB
5 Mbp	1 GB
200 Mbp	32 GB
3 Gbp	128 GB

Input file formats of ABySS

- FASTA
- FASTQ
- Illumina QSEQ
- Eland export
- SAM
- BAM
- Compressed: gz, bz2, xz, tar

Running ABySS

- Assemble the paired-end reads in the file reads.fa
  - abyss-pe **name**=ecoli **k**=32 **n**=10 **in**=reads.fa
- Assemble the paired-end reads in the files reads_1.fa and reads_2.fa:
  - abyss-pe **name**=ecoli **k**=32 **n**=10 **in**='reads_1.fa reads_2.fa'

Running ABySS in parallel

- Run ABySS using eight threads
  - abyss-pe **np=8** name=ecoli k=32 n=10 in='reads_1.fa reads_2.fa'
- ABySS uses MPI (Message Passing Interface)
- MPI is implemented by many vendors
- Open MPI is an open-source implementation

Running ABySS in parallel on a cluster (SGE)

- Run ABySS on a cluster using 8 threads
  - **qsub -pe openmpi 8 -N ecoli**
    abyss-pe ~~np=8 name=ecoli~~ k=32 n=10 in='reads_1.fa reads_2.fa'
- abyss-pe uses the environment variables *JOB_NAME* and *NSLOTS* passed to it by SGE as the default values for *name* and *np*

Running ABySS in parallel on a cluster (SGE) for many values of k

- Assemble every 8th k from 32 to 96
  - qsub -pe openmpi 8 -N ecoli **-t 32-96:8**
    abyss-pe ~~k=32~~ n=10
    in='reads_1.fa reads_2.fa'
- abyss-pe uses the environment variable *SGE_TASK_ID* passed to it by SGE as the default value for *k*

Assembling multiple libraries

- abyss-pe name=ecoli
  k=32 n=10
  lib='pe200 pe500'
  pe200='pe200_1.fa pe200_2.fa'
  pe500='pe500_1.fa pe500_2.fa'

Assembling a mix of paired-end and single-end reads

- abyss-pe name=ecoli
  k=32 n=10
  lib='pe200 pe500'
  pe200='pe200_1.fa pe200_2.fa'
  pe500='pe500_1.fa pe500_2.fa'
  se='long.fa'

Parameters of ABySS

- **name**: name of the assembly
- **lib**: names of the paired-end libraries
- **se**: paths of the single-end read files
- Example
  abyss-pe **name**=ecoli **k**=32 **n**=10
  lib='pe200 pe500'
  pe200='pe200_1.fa pe200_2.fa'
  pe500='pe500_1.fa pe500_2.fa'
  se='long.fa'

Parameters of ABySS Sequence assembly

- **k**: the size of a k-mer
- **q**: quality trimming removes low-quality bases from the ends of reads
- **e** and **c**: coverage-threshold parameters
  - e: erosion removes bases from the ends of contigs
  - c: coverage threshold removes entire contigs
- **p**: the minimum identity required for removing sequence variants (popping bubbles)

Optimizing k with a cluster

- Assemble every 8th k from 32 to 96
  Nine assemblies: 32 40 48 56 64 72 80 88 96
- Find the peak N50 (a measure of contiguity)
- Assemble every 2nd k around the peak N50
  For example, if the peak were at k=64...
  Eight assemblies: 56 58 60 62 66 68 70 72
- SGE:
  qsub -t 32-96:8 qsub-abyss.sh
  qsub -t 56-72:2 qsub-abyss.sh

Parameters of ABySS Paired-end assembly

- **s**: the minimum size of a seed contig
- **n**: the number of pairs required to join two contigs
- Example
  abyss-pe **name**=ecoli
  k=64 **q**=3 **p**=0.9 **s**=100 **n**=10
  lib='pe200 pe500'
  pe200='pe200_1.fa pe200_2.fa'
  pe500='pe500_1.fa pe500_2.fa'
  se='long.fa'

Output files of ABySS

- ${name}-**contigs.fa**
  The final contigs in FASTA format
- ${name}-**bubbles.fa**
  The equal-length variant sequences (FASTA)
- ${name}-**indel.fa**
  The different-length variant sequences (FASTA)
- ${name}-**contigs.dot**
  The contig overlap graph in Graphviz format

Intermediate output files of ABySS

- ${name}-**3.fa**
  The initial contigs in FASTA format
- .**adj**: contig overlap graph in ABySS adj format
- .**dist**: estimates of the distance between contigs in ABySS dist format
- .**path**: lists of contigs to be merged
- .**hist**: fragment-size histogram of a library
- **coverage.hist**: k-mer coverage histogram

Assembly/alignment visualization

Assembly/alignment visualization

- Display where the reads are placed in the assembly (or align to the reference)
- Show paired-end reads and highlight locations where the pairs are discordant
- Highlight discrepant sequence and variants
- Browse annotations
- Standard file formats are BAM, VCF and GFF, though there are many

Visualization tools

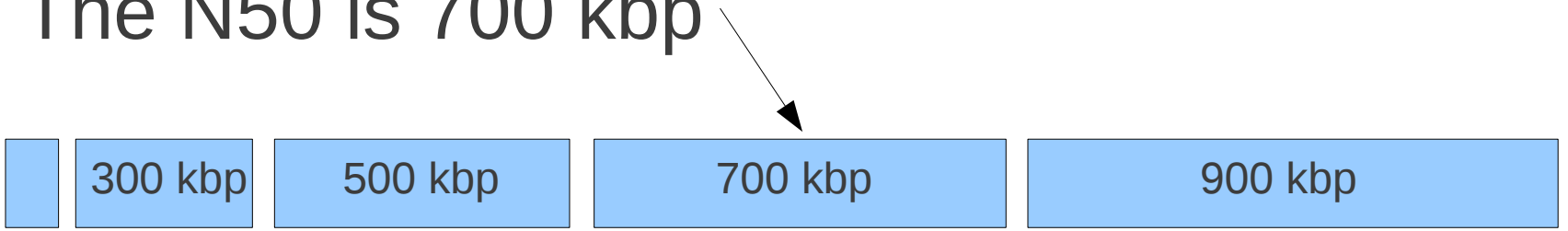
- UCSC Genome Browser
- Integrative Genomics Viewer (IGV)
- Tablet
- gap5
- consed

Integrative Genomics Viewer (IGV)

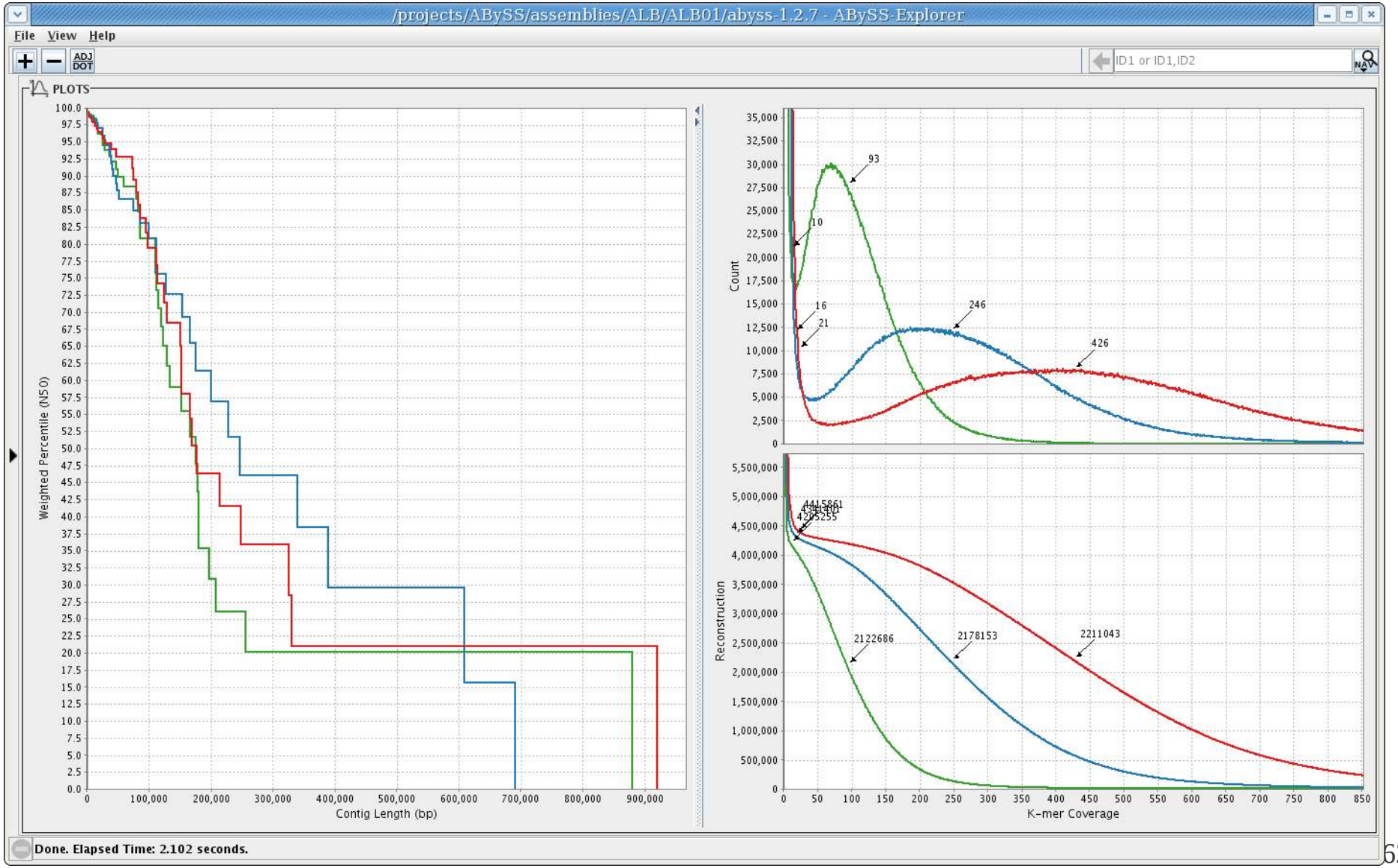
ABySS-Explorer

N50

- The N50 is the weighted median of the lengths of the contigs
- Example: an assembled genome in five contigs 100, 300, 500, 700 and 900 kbp
- The genome is 2500 kbp
- Half the genome is assembled in two contigs: 700 and 900 kbp
- The N50 is 700 kbp

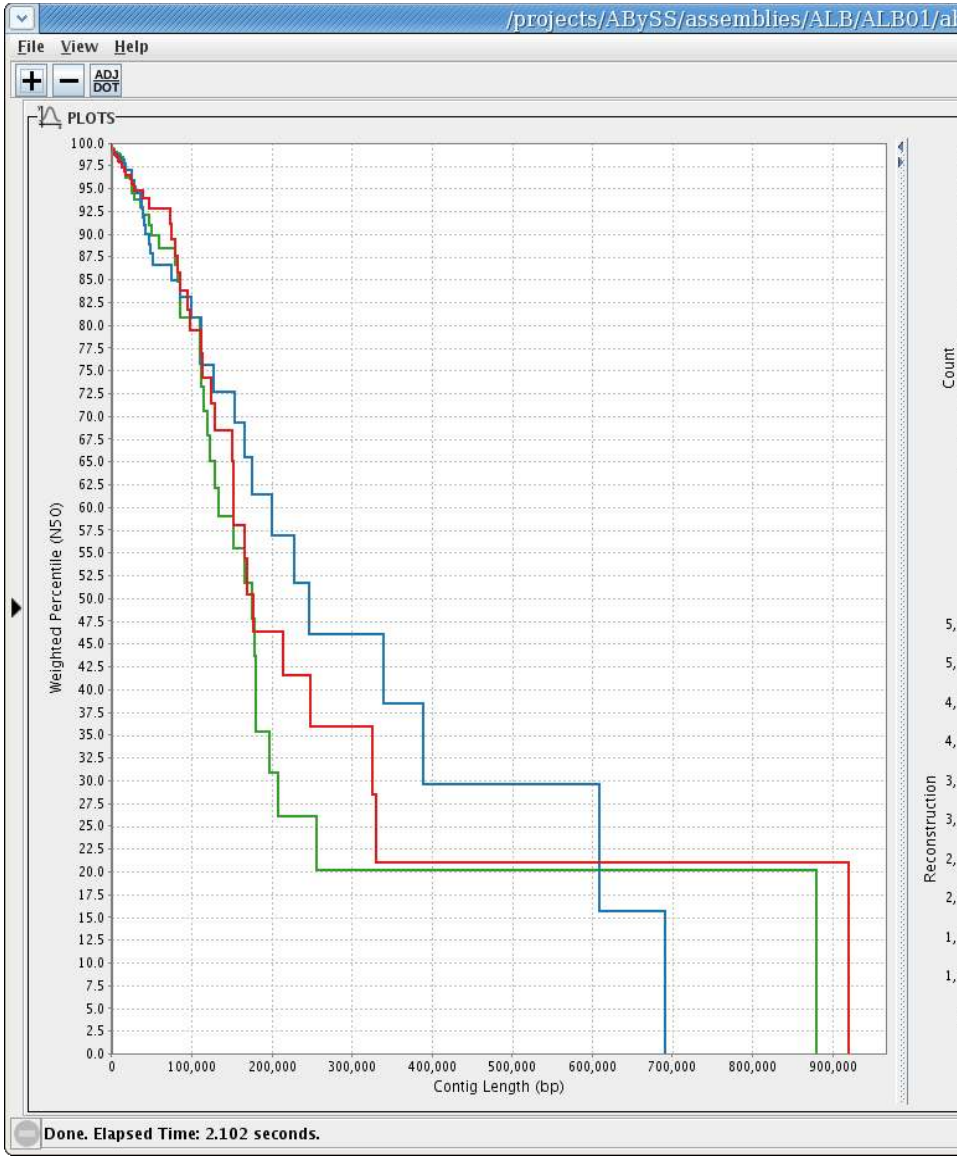


ABySS-Explorer



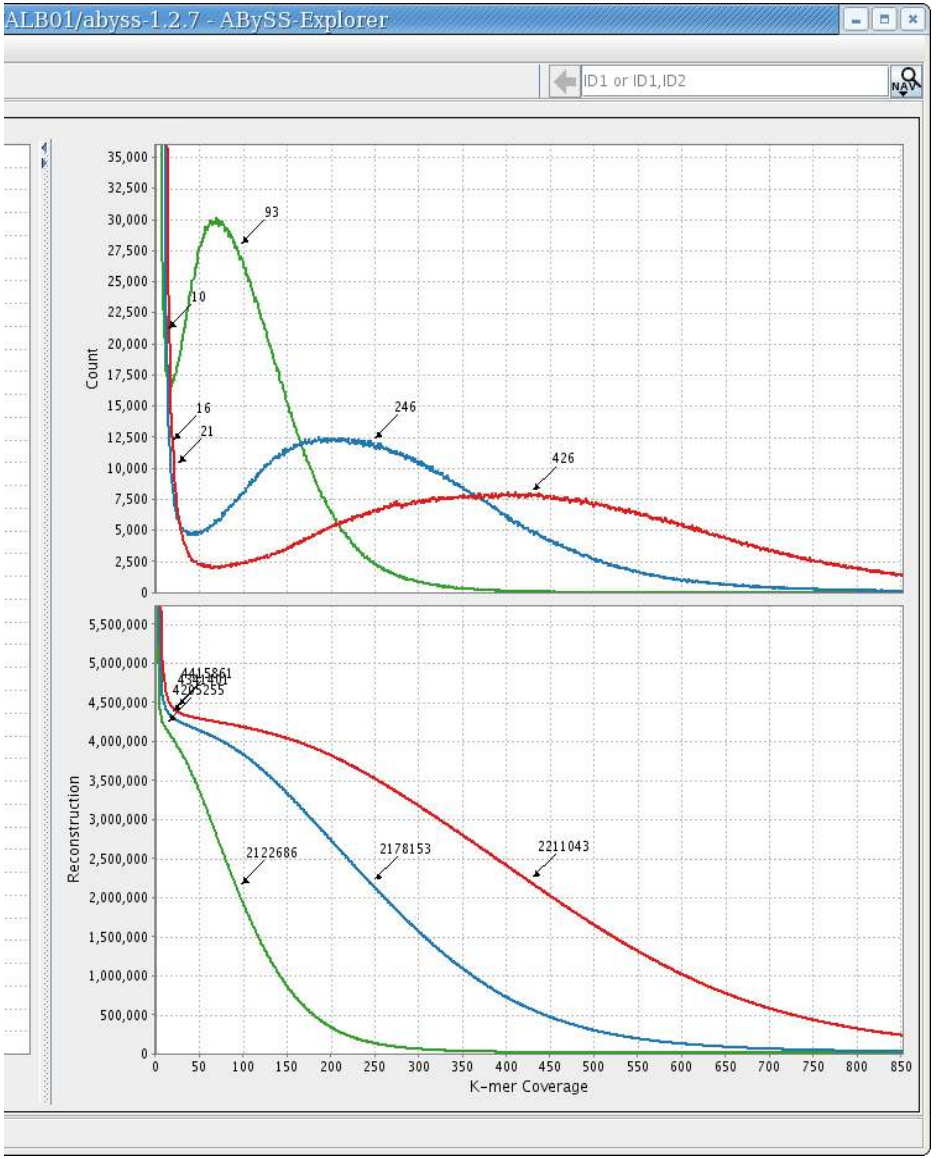
N50 and Nxx plot

- The N50 is the weighted median (50th percentile) of contig sizes
- The N50 summarizes a single point of the Nxx plot
- Better assemblies are further to the right



K-mer coverage histogram

- Counts the number of occurrences of each k-mer
- Useful for estimating the size of the genome



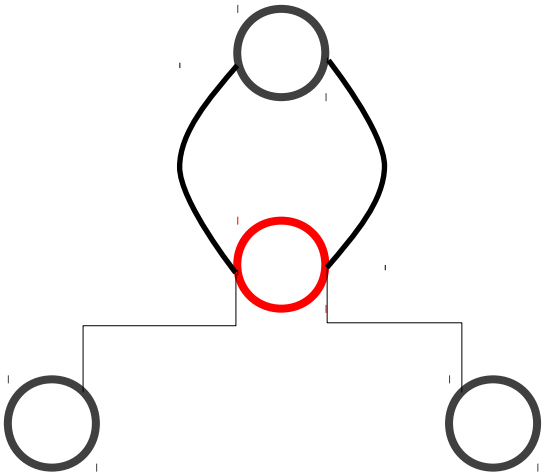
ABySS-Explorer
Assembly graph visualization

Assembly Ambiguities

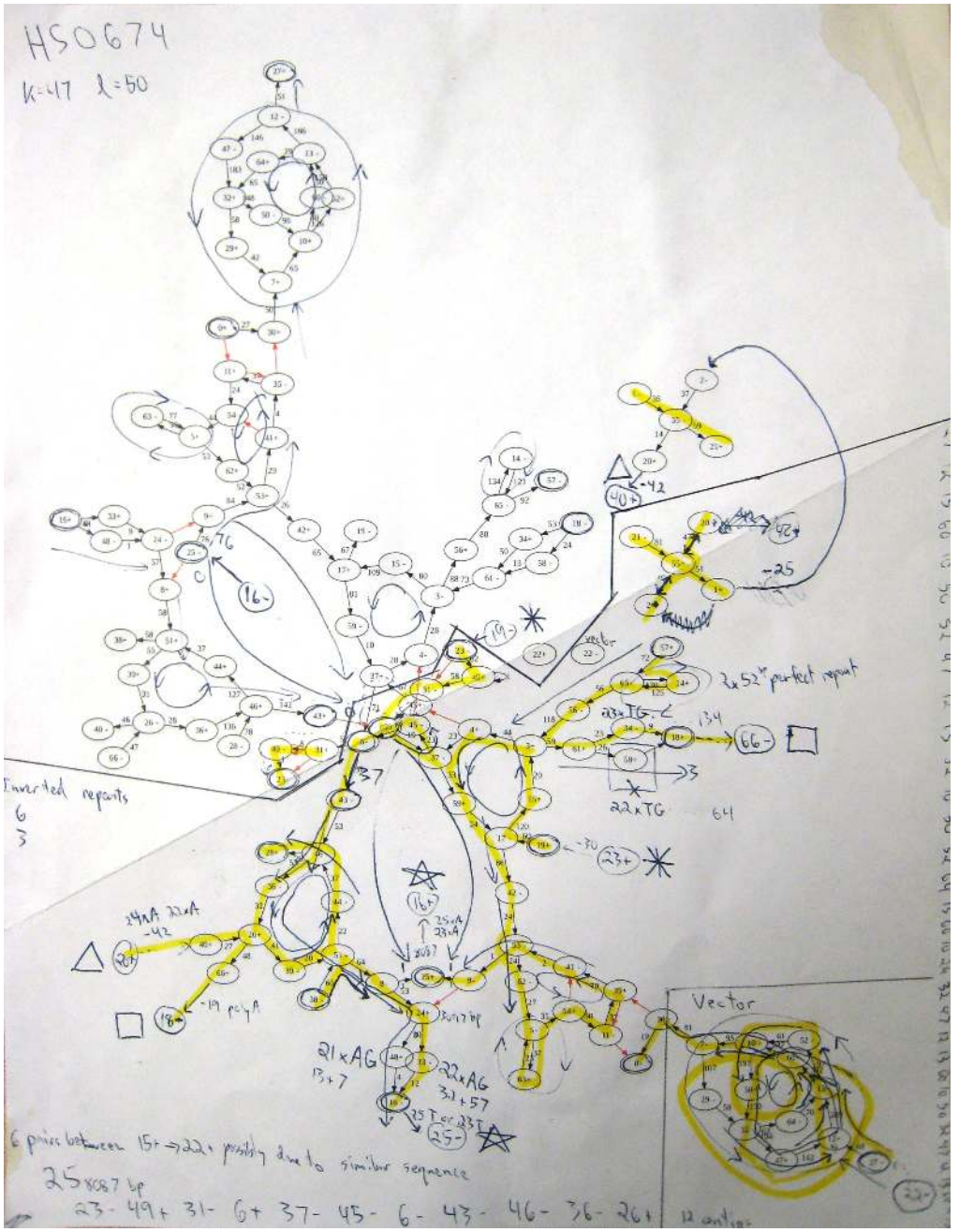
True genome sequence

GGATTGAAAAAAAAAAAAAAAAGTAGCACGAATATACATAGAAAAAAAAAAAAAAAAATTACG

Assembled sequence
de Bruijn graph representation



Starting Point



5'-CTCTCTCTTCCAGTAAGACTGCGAAAAAT-3'

5'-AAAAATGCCAGTAAGACTGACGGGGGG-3'

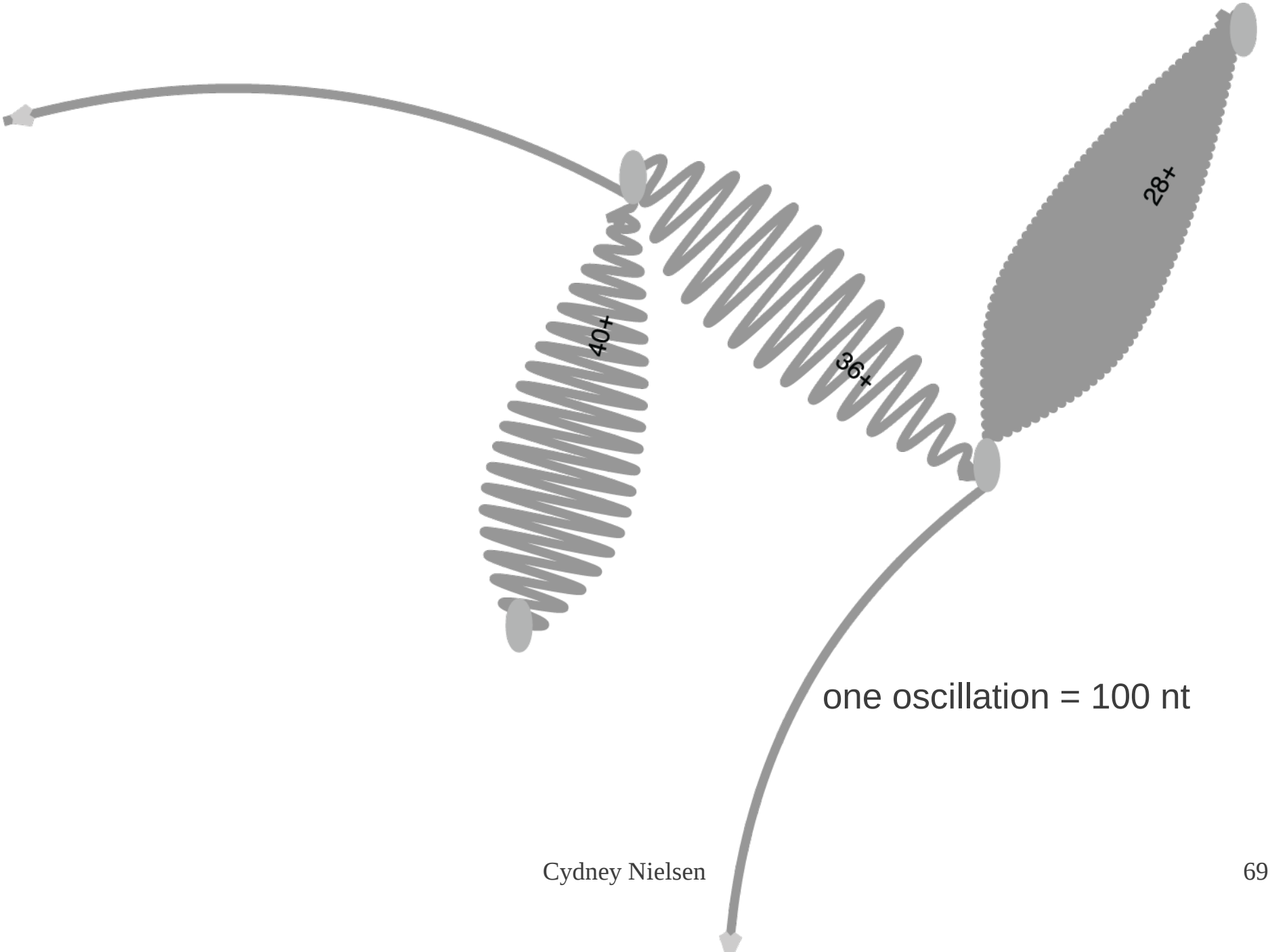
5'-CTCTCTCTTCCAGTAAGACTGCGAAAAAT-3'

3'-TTTTTACGGTCATTCTGACTGCCCCCC-5'

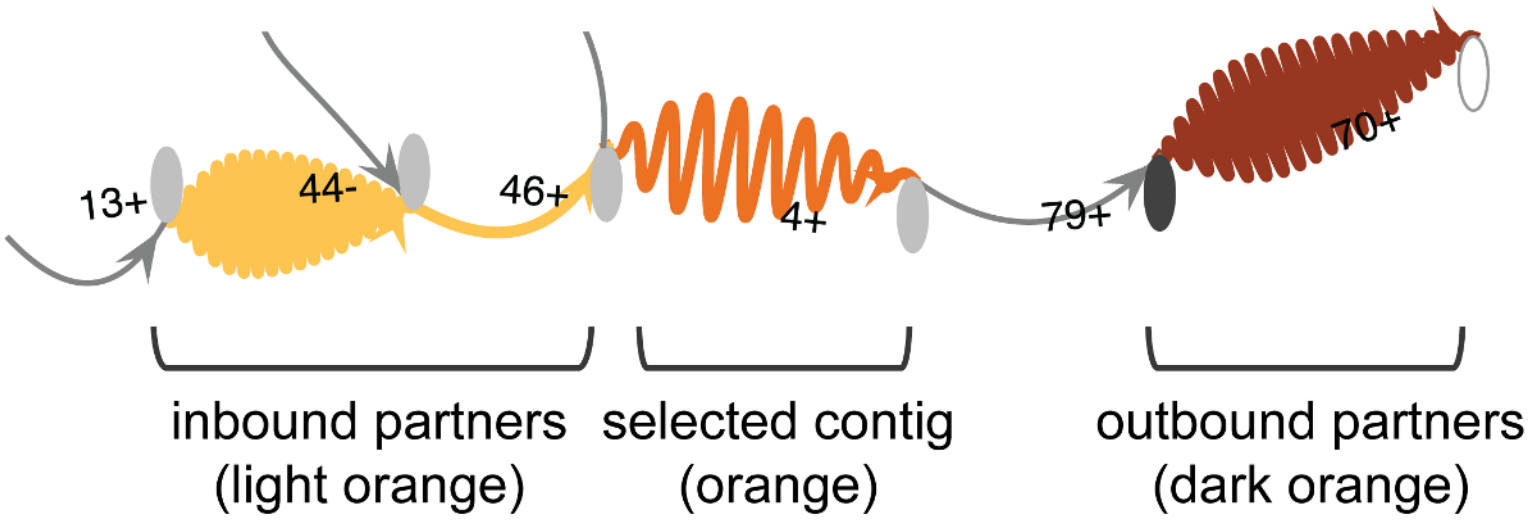
3'-TCTCTCGAAGGTCATTCTGACGCTTTTTA-5'

3'-TTTTTACGGTCATTCTGACTGCCCCCC-5'

Sequence length



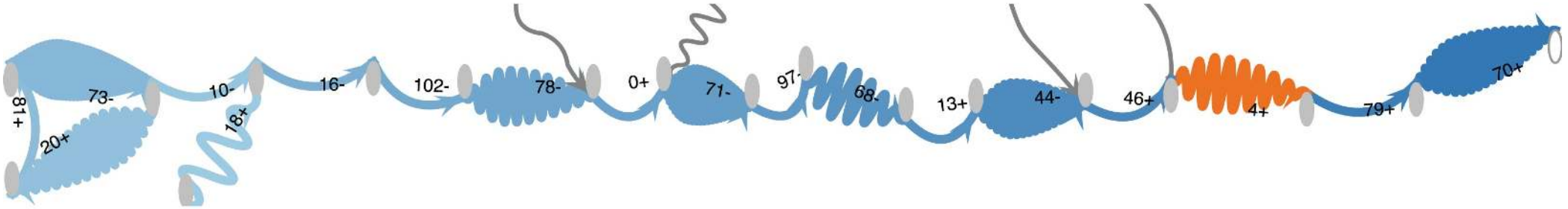
Paired-end reads



After building the initial single-end (SE) contigs from *k*-mer sequences, ABySS uses paired-end reads to resolve ambiguities.

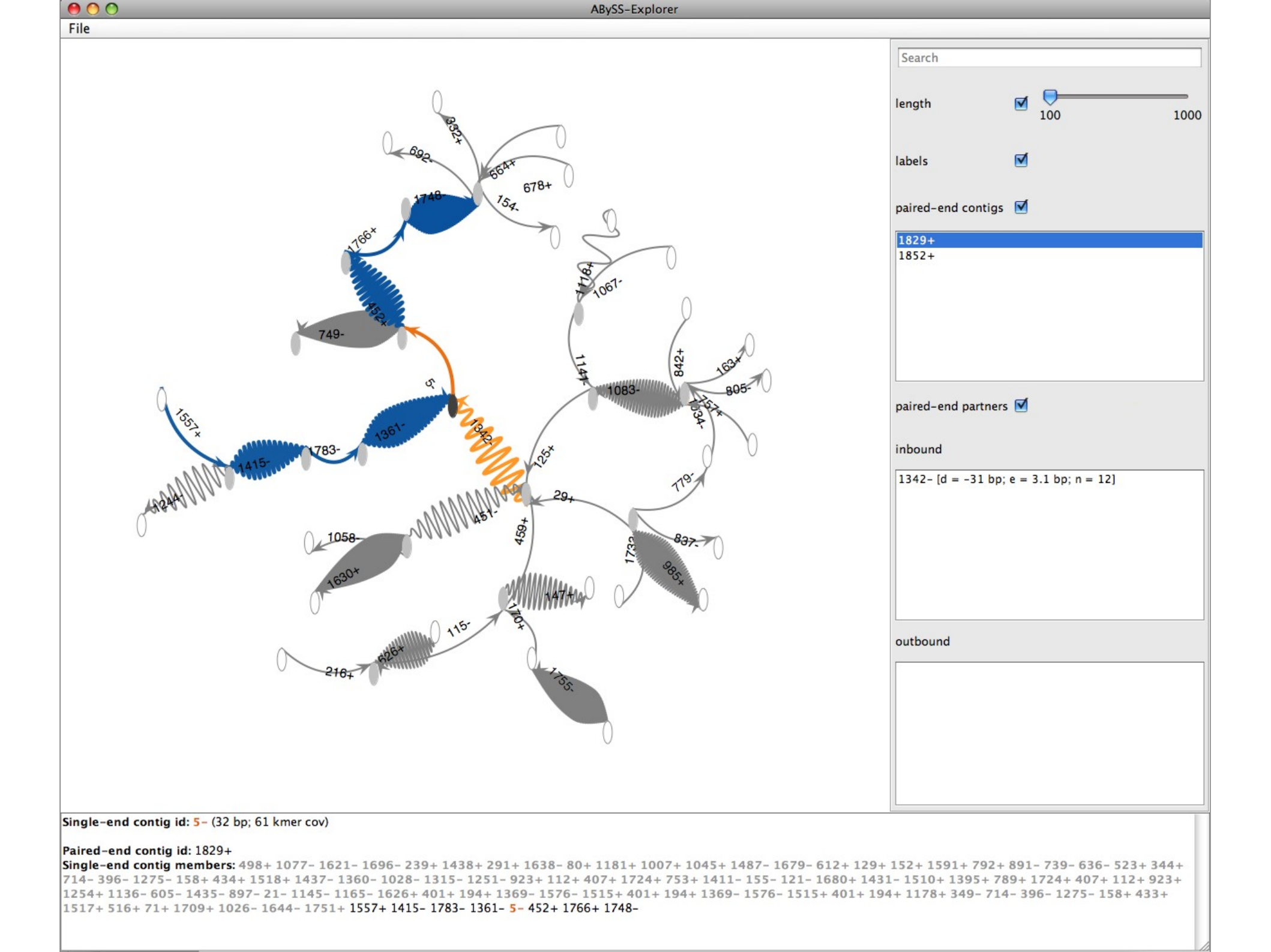
Paired-end contigs

Paired-end reads are used to construct paired-end (PE) contigs



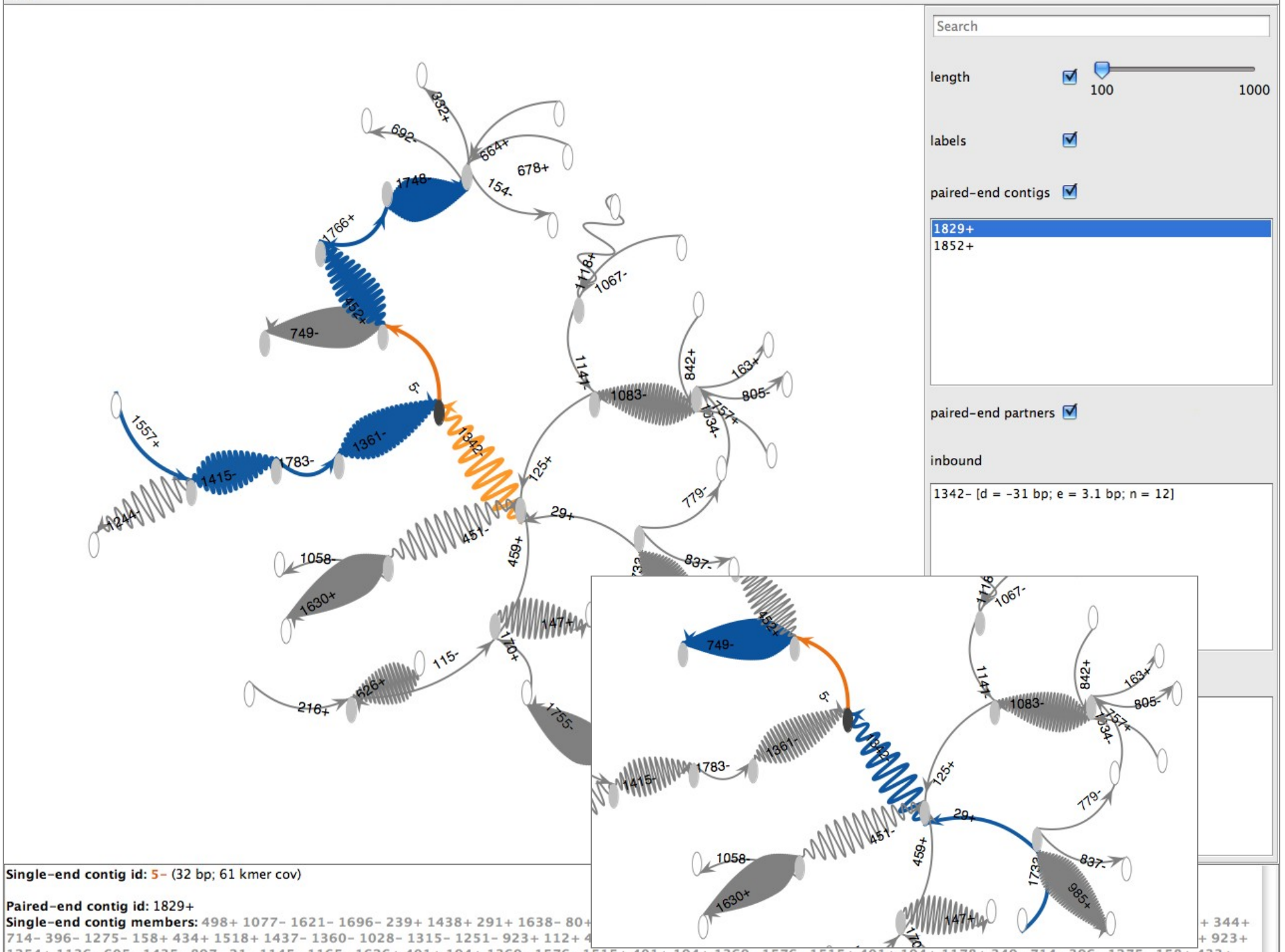
… **13+ 44- 46+ 4+ 79+ 70+** …

blue gradient = paired end contig
orange = selected single end contig

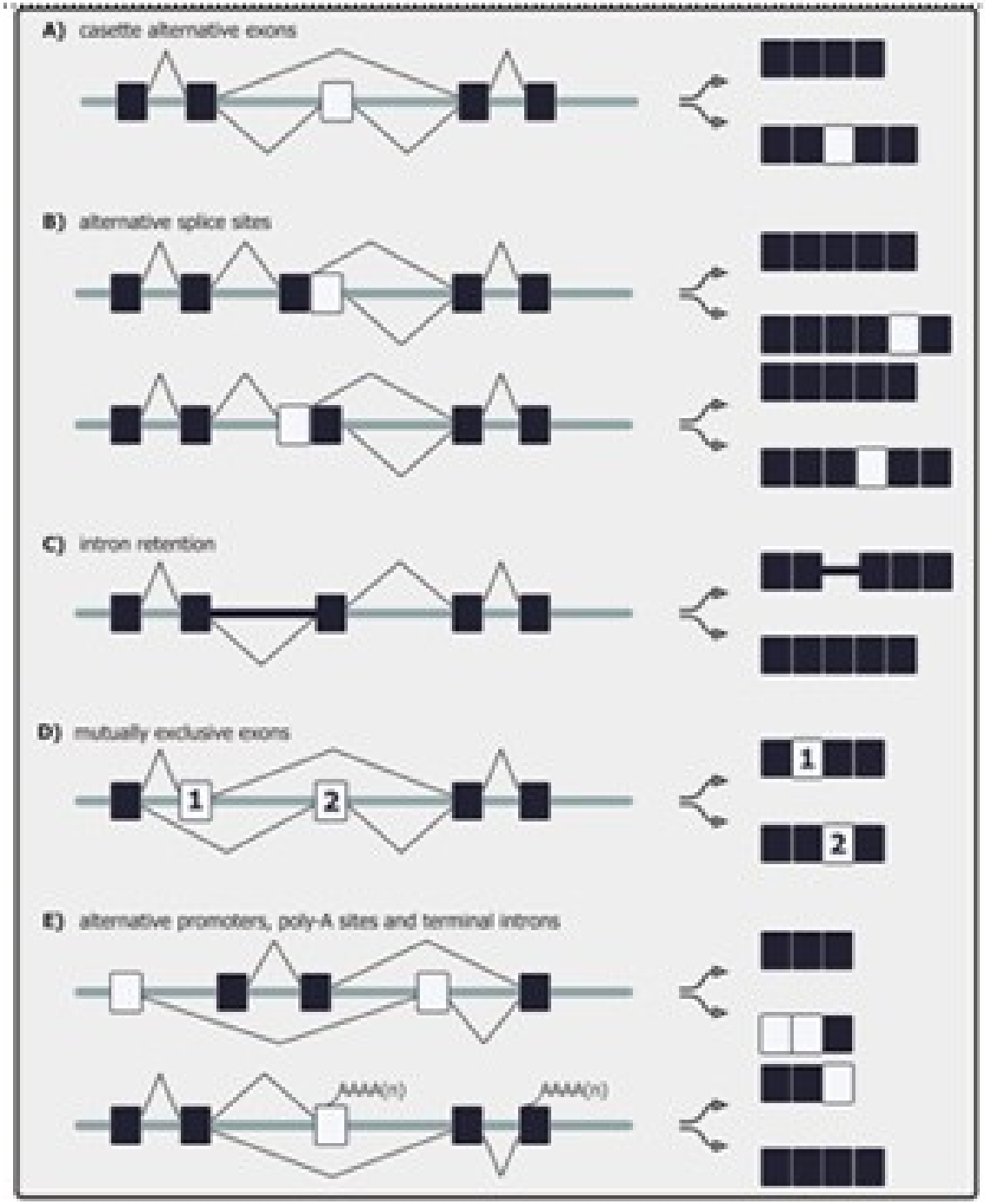

ABySS-Explorer
File
Search
length
100
1000
labels
paired-end contigs
1829+
1852+
paired-end partners
inbound
1342- [d = -31 bp; e = 3.1 bp; n = 12]
outbound
Single-end contig id: 5- (32 bp; 61 kmer cov)
Paired-end contig id: 1829+
Single-end contig members: 498+ 1077- 1621- 1696- 239+ 1438+ 291+ 1638- 80+ 1181+ 1007+ 1045+ 1487- 1679- 612+ 129+ 152+ 1591+ 792+ 891- 739- 636- 523+ 344+ 714- 396- 1275- 158+ 434+ 1518+ 1437- 1360- 1028- 1315- 1251- 923+ 112+ 407+ 1724+ 753+ 1411- 155- 121- 1680+ 1431- 1510+ 1395+ 789+ 1724+ 407+ 112+ 923+ 1254+ 1136- 605- 1435- 897- 21- 1145- 1165- 1626+ 401+ 194+ 1369- 1576- 1515+ 401+ 194+ 1369- 1576- 1515+ 401+ 194+ 1178+ 349- 714- 396- 1275- 158+ 433+ 1517+ 516+ 71+ 1709+ 1026- 1644- 1751+ 1557+ 1415- 1783- 1361- 5- 452+ 1766+ 1748-

ABySS-Explorer

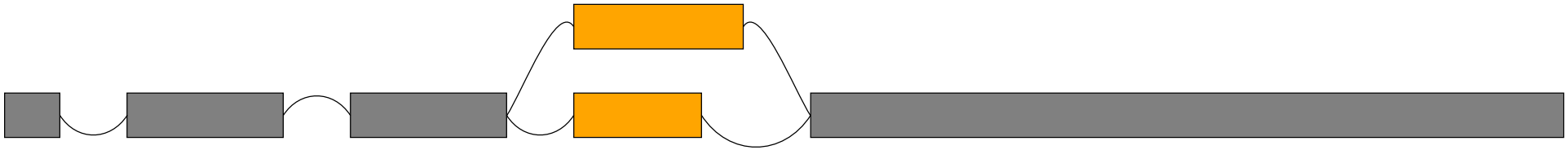
File



Transcriptome Assembly, Alternative Splicing and Visualization



http://www.eurasnet.info/clinicians/alternative-splicing/what-is-alternative-splicing/diversity

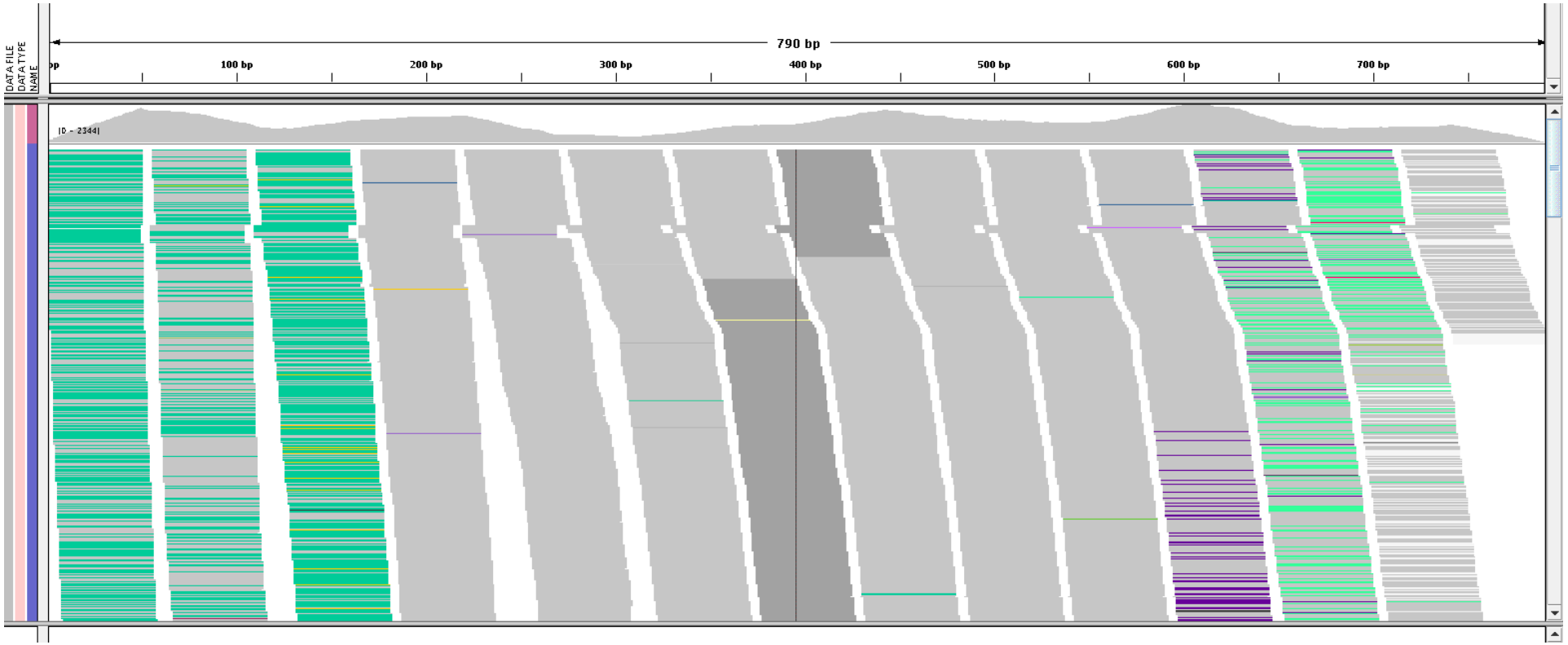


Assembly	ABySS
Alignment	GMAP
Detection & Visualisation	Sircah

ABySS

Assemble transcriptome data

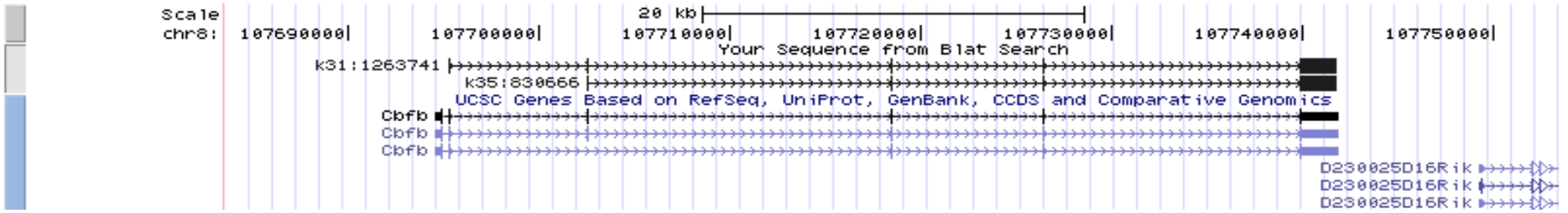
Transcriptome reads → Assembly

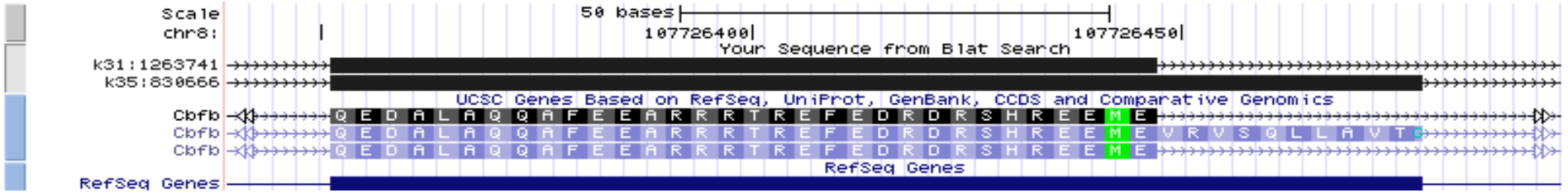

DATA FILE
DATA TYPE
NAME
790 bp
100 bp
200 bp
300 bp
400 bp
500 bp
600 bp
700 bp
ID - 23441

GMAP

Align contigs to the reference genome
Annotate introns

Assembly → Alignments

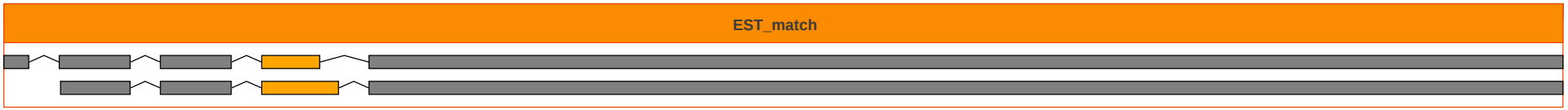

Scale
chr8:
20 kb
107690000
107700000
107710000
107720000
107730000
107740000
107750000
Your Sequence from Blat Search
k31:1263741
k35:830666
UCSC Genes Based on RefSeq, UniProt, GenBank, CCDS and Comparative Genomics
Cbfb
Cbfb
Cbfb
D230025D16Rik
D230025D16Rik
D230025D16Rik


Scale
chr8:
50 bases
107726400
107726450
Your Sequence from Blat Search
k31:1263741
k35:830666
UCSC Genes Based on RefSeq, UniProt, GenBank, CCDS and Comparative Genomics
Cbfb
Cbfb
Cbfb
RefSeq Genes
RefSeq Genes

Sircah

Detect alternative splicing events

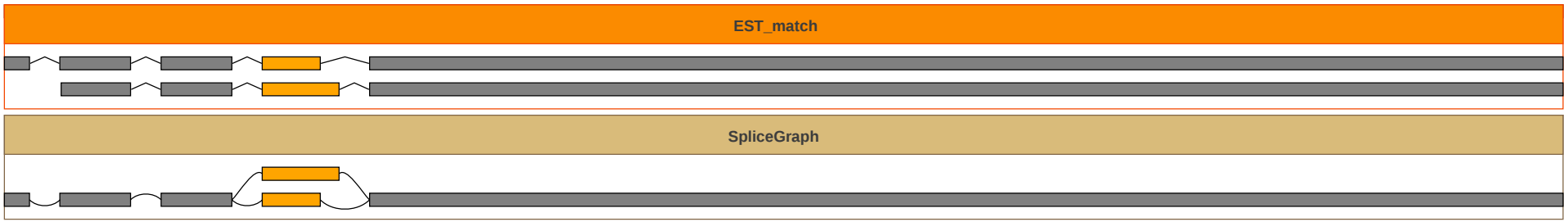
Alignments → Alternative splicing


EST_match

Sircah Visualisation

Draw splicing diagrams

Alternative splicing → Splicing diagrams


EST_match
SpliceGraph

Acknowledgments

Supervisors

- İnanç Birol
- Steven Jones

Team

- Readman Chiu
- Rod Docking
- Ka Ming Nip
- Karen Mungall
- Jenny Qian
- Tony Raymond

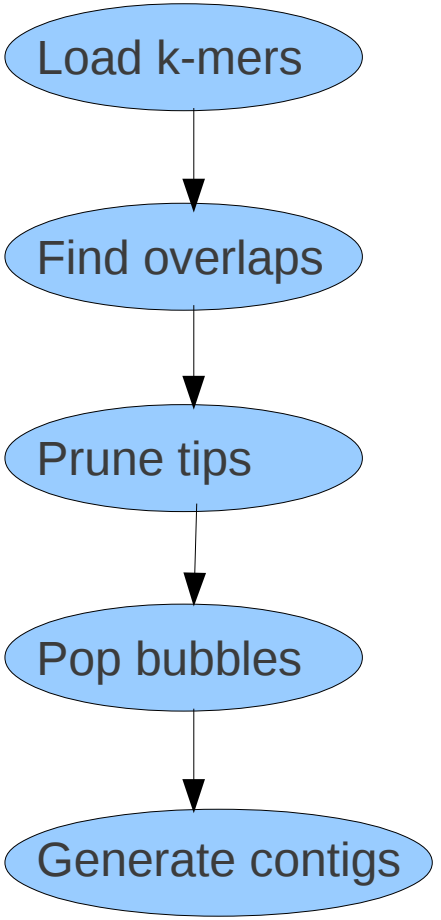
ABySS Algorithm

An assembly in two stages

- Stage I: Sequence assembly algorithm
- Stage II: Paired-end assembly algorithm

Stage 1
Sequence assembly algorithm

- Load the reads,
  breaking each read into k-mers
- Find adjacent k-mers, which
  overlap by k-1 bases
- Remove k-mers resulting from
  read errors
- Remove variant sequences
- Generate contigs



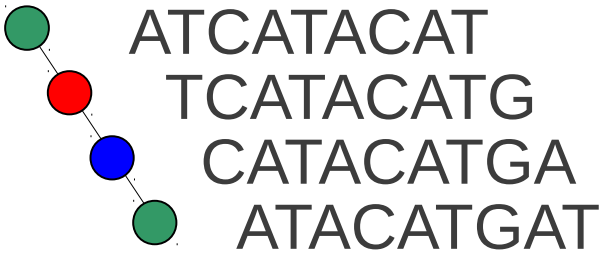
Load the reads

- For each input read of length l, (l - k + 1) k-mers are generated by sliding a window of length k over the read

Read (l = 12):

ATCATACATGAT

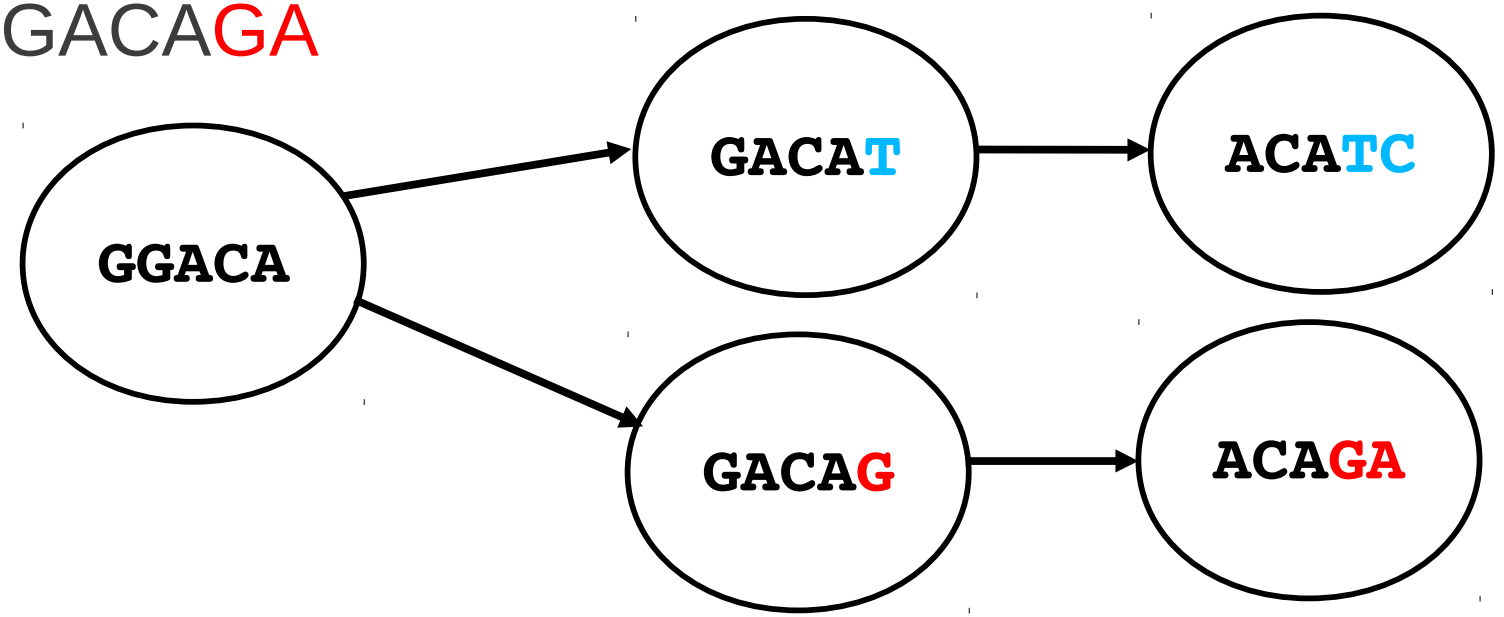
k-mers (k = 9):



- Each k-mer is a vertex of the de Bruijn graph
- Two adjacent k-mers are an edge of the de Bruijn graph

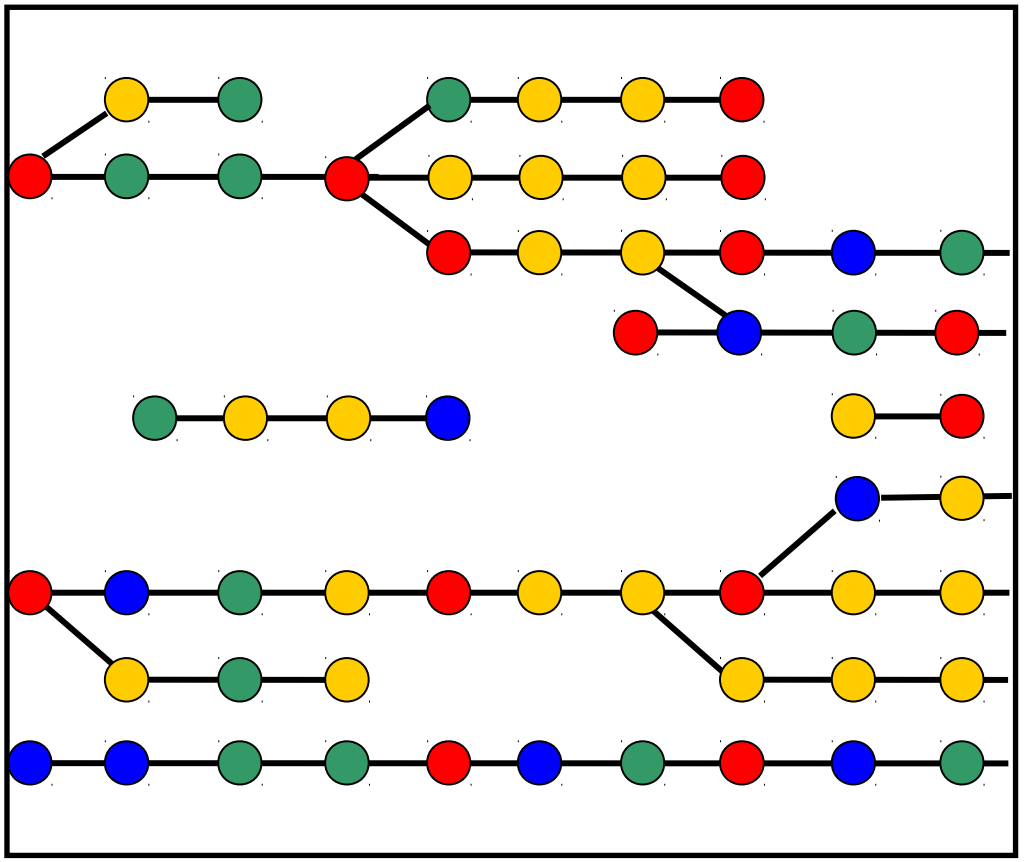
De Bruijn Graph

- A simple graph for k = 5
- Two reads
  - GGACATC
  - GGACAGA



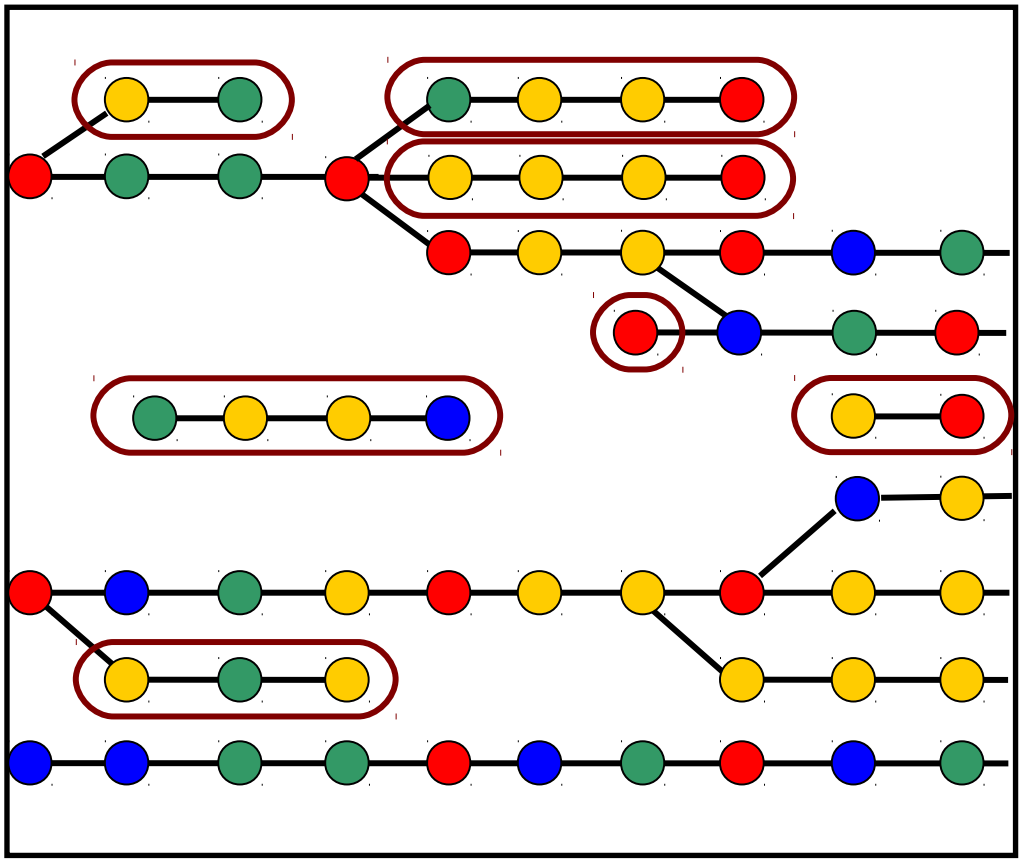
Pruning tips

- Read errors cause tips



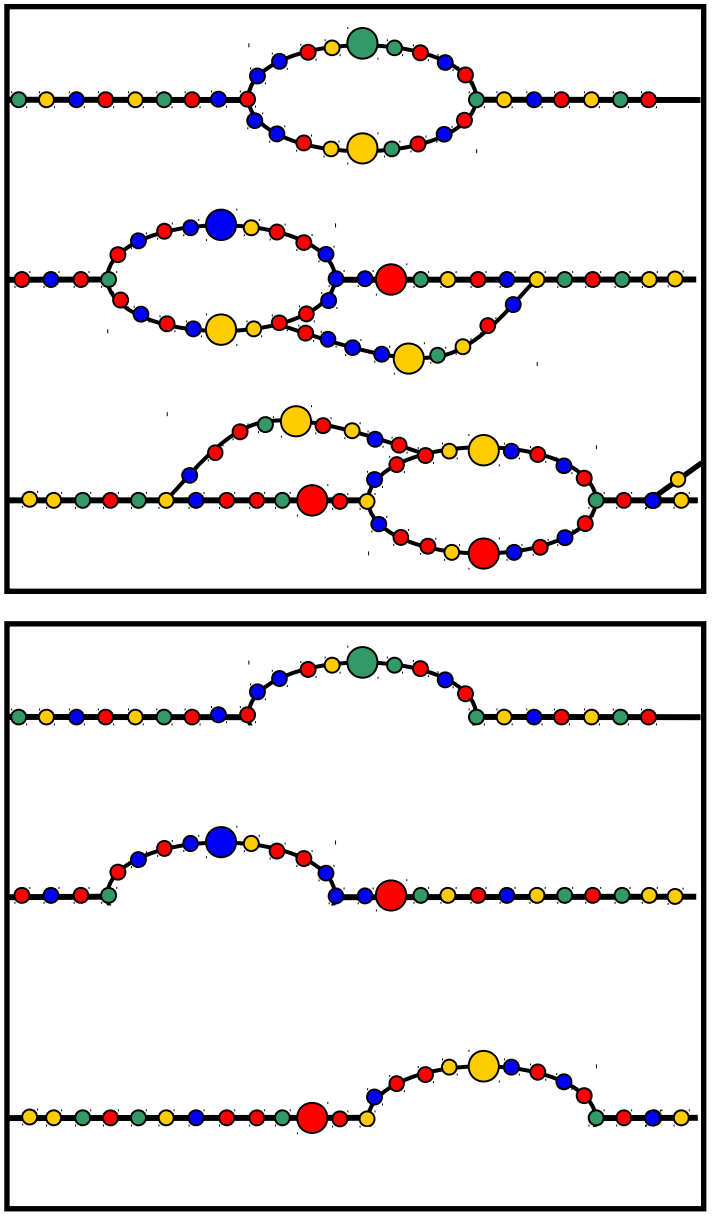
Pruning tips

- Read errors cause tips
- Pruning tips removes the erroneous reads from the assembly



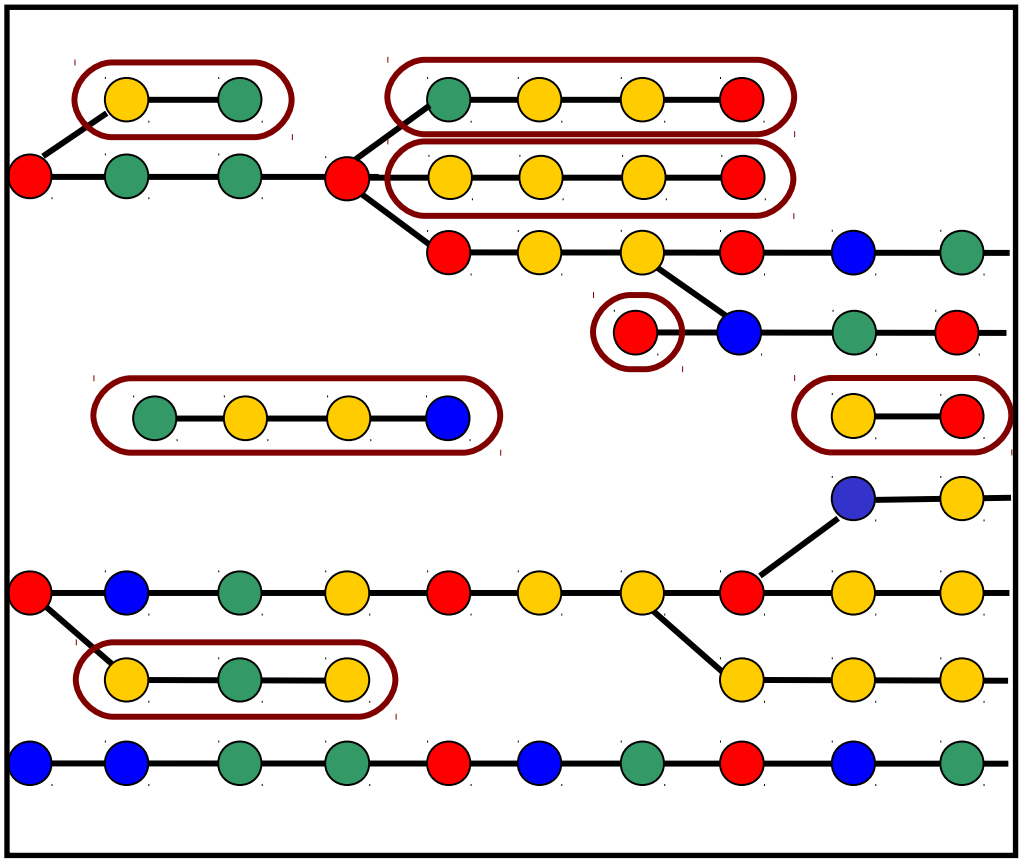
Popping bubbles

- Variant sequences cause bubbles
- Popping bubbles removes the variant sequence from the assembly
- Repeat sequences with small differences also cause bubbles



Assemble contigs

- Remove ambiguous edges
- Output contigs in FASTA format



Paired-end assembly algorithm
Stage 2

- Align the reads to the contigs of the first stage
- Generate an empirical fragment-size distribution using the paired reads that align to the same contig
- Estimate the distance between contigs using the paired reads that align to different contigs

Align the reads to the contigs

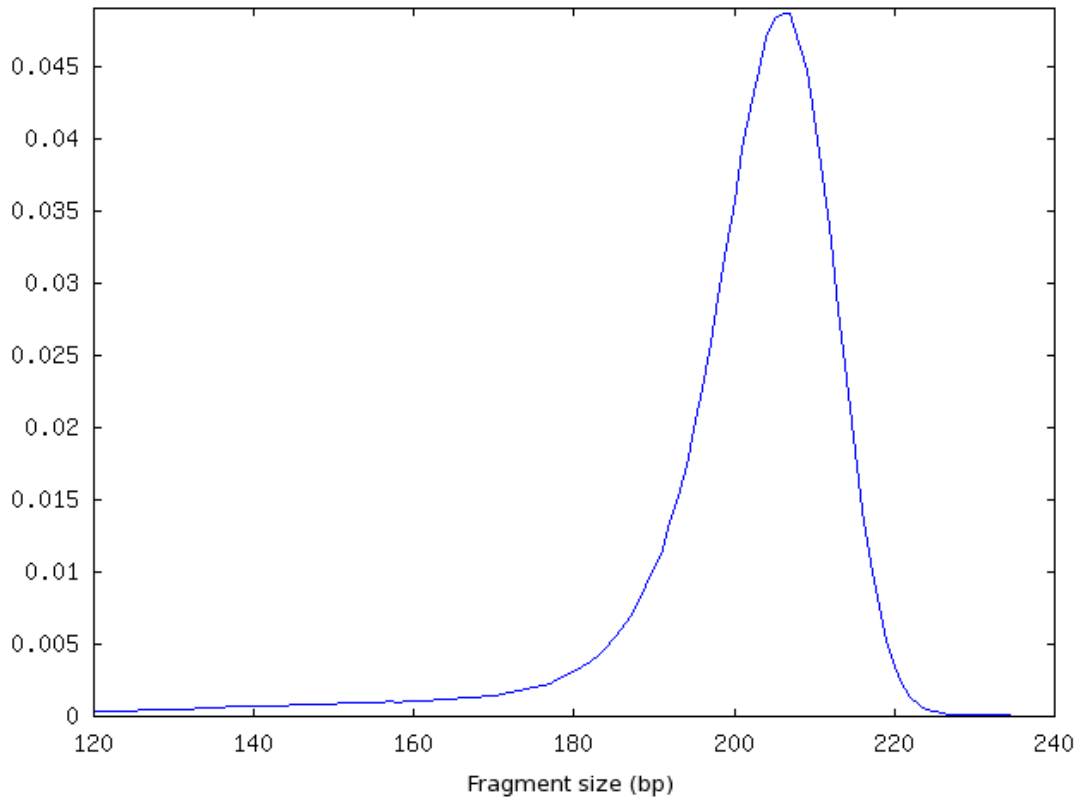
KAligner

- Every k-mer in the single-end assembly is unique
- KAligner can map reads with k consecutive correct bases
- ABySS may use other aligners, including BWA and bowtie

Empirical fragment-size distribution

ParseAligns

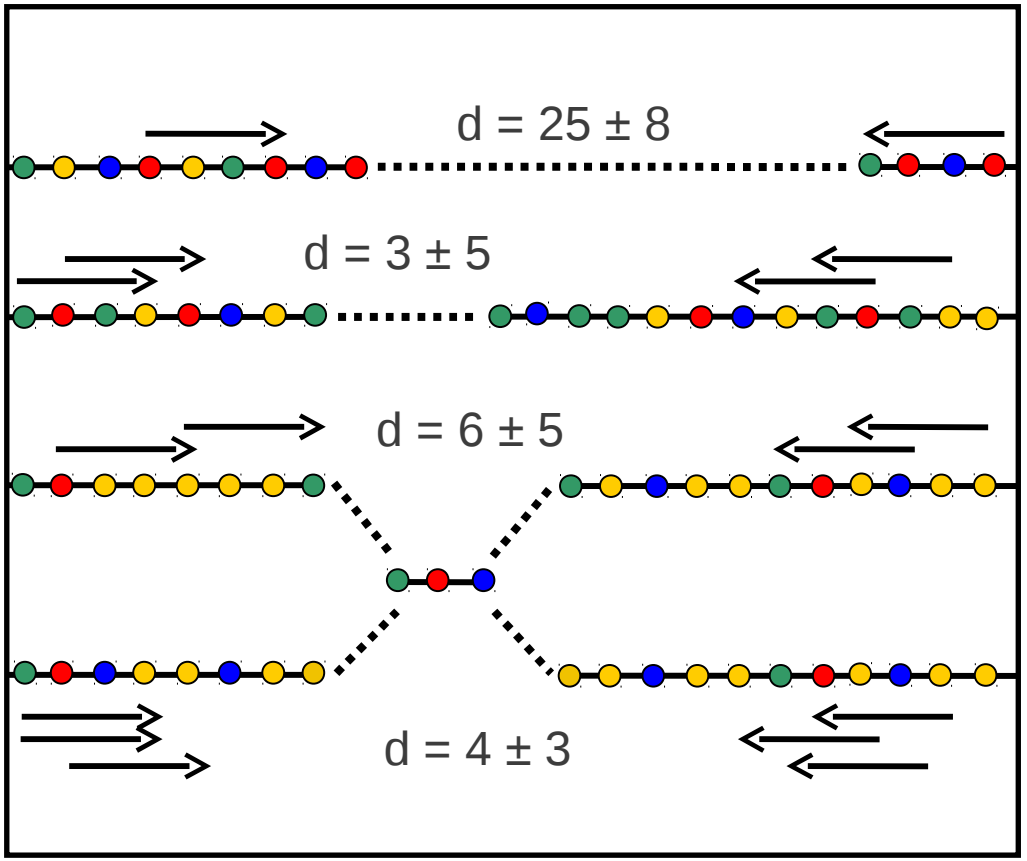
- Generate an empirical fragment-size distribution using the paired reads that align to the same contig



Estimate distances between contigs

DistanceEst

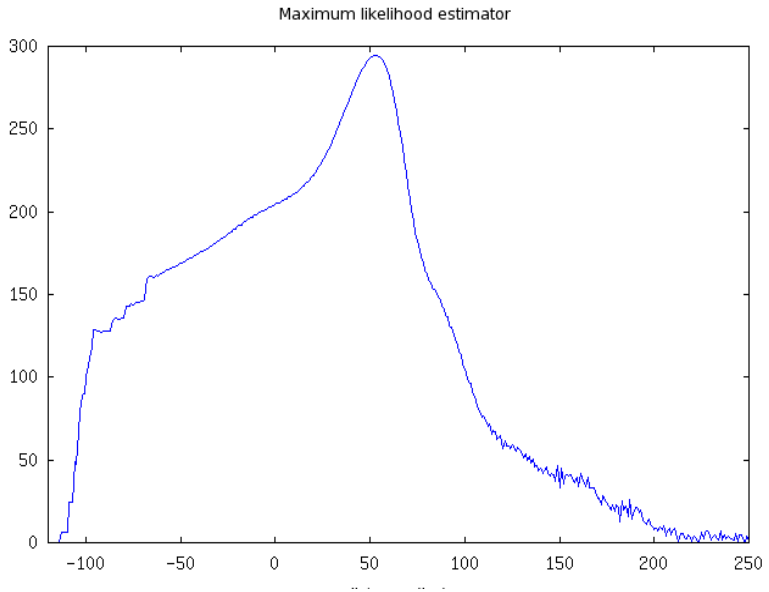
- Estimate the distance between contigs using the paired reads that align to different contigs

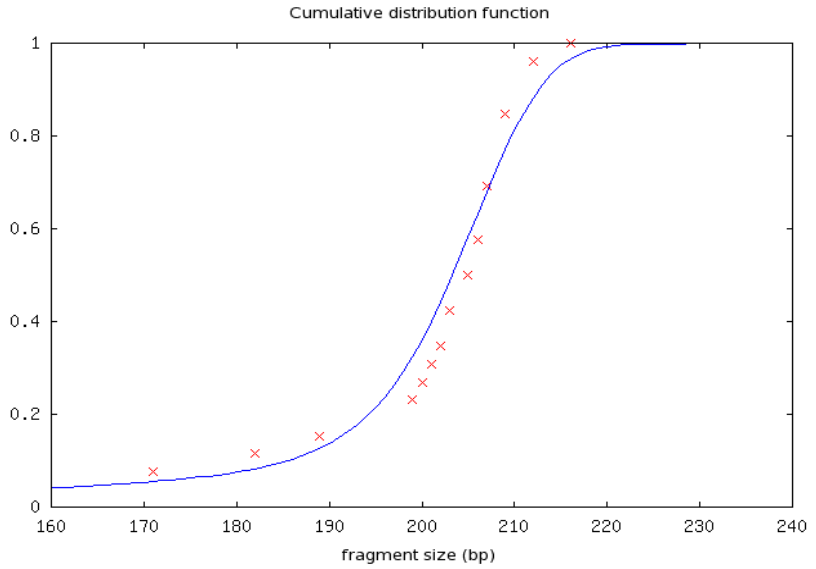


Maximum likelihood estimator
DistanceEst

- Use the empirical paired-end size distribution
- Maximize the likelihood function
- Find the most likely distance between the two contigs

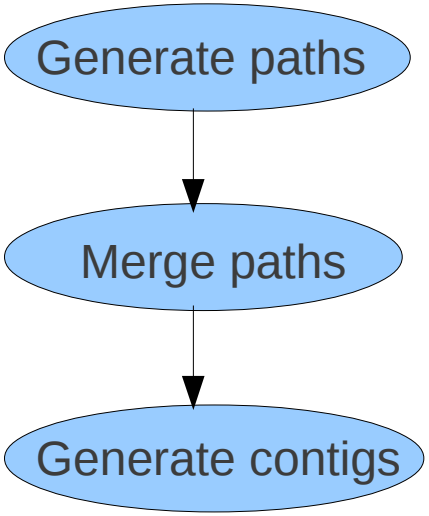
$$\mathcal{L}(\theta) = \prod_{i=1}^{n} f_\theta(x_i)$$





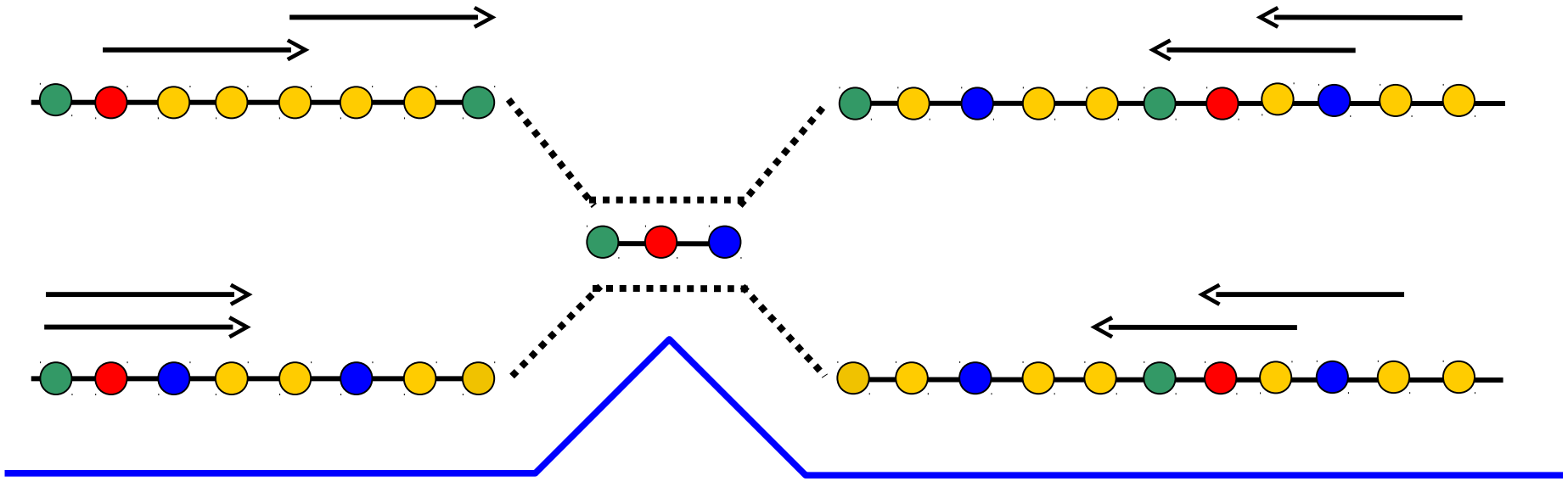
Paired-end algorithm
continued...

- Find paths through the contig adjacency graph that agree with the distance estimates
- Merge overlapping paths
- Merge the contigs in these paths and output the FASTA file



Find consistent paths
SimpleGraph

- Find paths through the contig adjacency graph that agree with the distance estimates

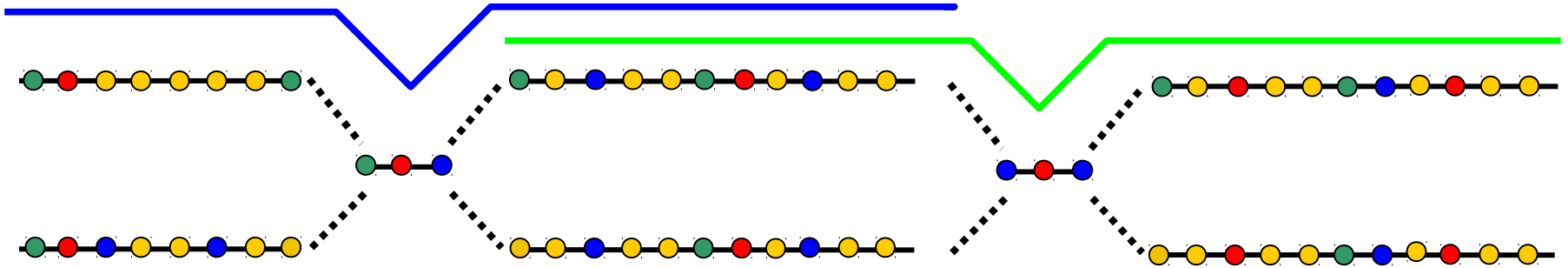


d = 4 ± 3

Actual distance = 3

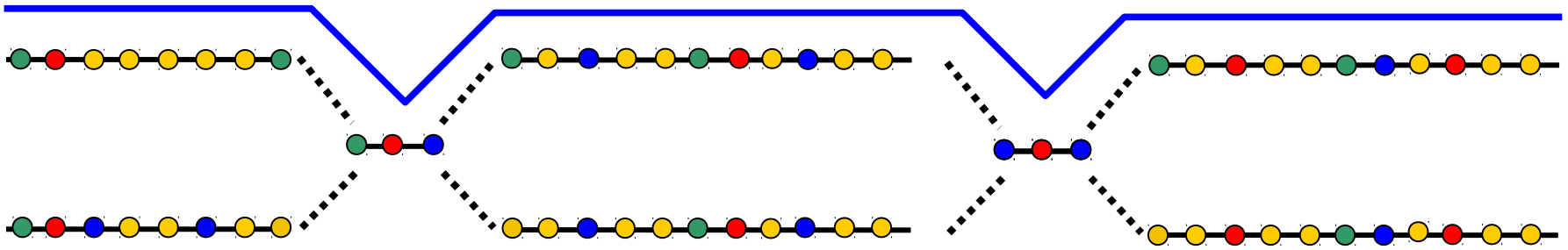
Merge overlapping paths
MergePaths

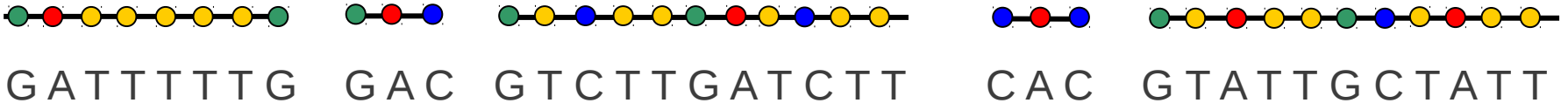
- Merge paths that overlap



Generate the FASTA output

- Merge the contigs in these paths.
- Output the FASTA file





Acknowledgments

Supervisors

- İnanç Birol
- Steven Jones

Team

- Readman Chiu
- Rod Docking
- Ka Ming Nip
- Karen Mungall
- Jenny Qian
- Tony Raymond